|  |
| --- |
| Accenture |
| Create wbs user guide |
|  |

# Introduction

This document aims at providing a simple user guide for the create\_wbs script. The document is internal to Accenture Mobility Services. It is targeted at software developers maintaining the script and project managers managing project plans in RTC.

# Scope

In RTC, work items can be linked through a parent child relationship, allowing for pieces of work to be broken down in smaller refined work items. This is referred to as the work breakdown structure (WBS). Unfortunately the exported file does not visually capture this relationship; thus making it difficult to navigate the work items. The create\_wbs script enhances exported data by sorting the work items and grouping them in a work break down structure.

# Prerequisites

Create\_wbs is written in python and json and is available from the rtc\_create\_wbs git repository. Below are the software prerequisites for using this script:

1. Git installed
2. Access to the AMS HOF git infrastructure
3. RTC exported csv file with the following mandatory fields:
   1. Planned for
   2. Priority
   3. Rank (relative to Priority)
   4. Id
   5. Parent
   6. Type
   7. Story points

## Python

1. Download Python 3.3.5 from <https://www.python.org/download/releases/3.3.5/>
2. Follow installation process and remember the file path
3. Open Control Panel/System and Security/System/Advanced system settings
4. On the Advanced tab, click on Environment Variables
5. Click on System variables
6. Append your chosen path following Path (e.g ";c:\Python30" if you have installed on your c drive)

## Xlswriter

1. Download xlswriter from <http://xlsxwriter.readthedocs.org/getting_started.html#getting-started>
2. At command prompt, run command python setup.py install

## Git Repo

1. At command prompt, navigate to a folder where the scripts will be stored.
2. Pull down the scripts using the git command

git clone https://*[your user id]*@ams.accenture.com/gerrit/p/rtc\_wbs.git

1. Run the command python create\_wbs.py *[name of input csv file] [name of output file]*

# Voila

## Creating a Query in RTC

## Running RTC query/export to xlsx

## The output file

Create\_wbs.py generates an xlsx file with 3 tabs:

1. Input tab is populated with the raw data from the input file
2. Ranked tab is populated with the work items sorted on fields “Planned For”, “Priority”, “Rank” and “ID”
3. Grouped tab displayed the work breakdown structure with the child work items grouped under the parent work item.

## The config file

This script allows the user to configure the formatting and ranking order that is required for determining the grouping. This can be easily done through a Python file called config.py.

The config file contains comments to help you understand what the different configuration does. You can modify any part of the config file by commenting it out or changing it. You shouldn’t need a detailed (or even any) knowledge of Python to edit the file so long as you respect and copy the format of the original version. Watch out for commas in lists.

### Plans

The entry "Planned For" is a list consisting of the names of the project plans in their order or priority. The script sorts the plans in ascending order in the iteration report. It doesn’t use PlannedFor order for ranking in the Work Breakdown or Ranked sheets as RTC doesn’t use it here either. The name of the plans must correspond to the names in the RTC export and assigned with the correct priority:

"Planned For":[

“name of plan”,

],

### Priority

The “Priority” entry is a list consisting of priority names in descending priority. The script ranks the work items in descending (i.e same order as the list you give it.

"Priority":[

"High",

"Medium",

"Low",

"Unassigned"

]

Change this section if you change priority enumeration in the RTC. If you don’t change the enumeration, you’re unlikely to need to change this.

### Format

The “Format” entry is a dictionary consisting of configuration for the cells which require distinct text formatting in the output file.

entry :{<key>:<value>,<key>:<value>,…},

Any text matching *entry* will be formatted using the *<key>:<value>* formatting rules associated with it. Description of how to construct a formatting rule can be found here: <http://xlsxwriter.readthedocs.org/en/latest/working_with_formats.html>

Example formats are listed below:

"Capability":{"font":"Times New Roman","size":10},

"Defect":{"font":"Times New Roman","size":10},

"Epic":{"font":"Times New Roman","size":16,"italic":1},

"Feature":{"font":"Times New Roman","size":14},

"Story":{"font":"Times New Roman","size":12},

"Impeded":{"color":"red"},

"More Information":{"color":"orange"},

"Done":{"color":"green"},

"In Progress":{"color":"#00ff77"},

"Implementing":{"color":"#00ff77"},

"Implemented":{"color":"#00ffbb"}

In addition to these generic formatting rules, the follow special *<entry>* values are defined: "Header, "Error\_Item”, "Iteration\_Summary”, "Category\_Summary". These control the formatting of specific summary rows within the report. (Note: Iteration\_Summary and Category\_Summary are used in the Iteration Report sheet. In RTC: Iteration is synonymous with PlannedFor and Category maps to FiledAgainst and Teams)